Sploitego

Maltego's (Local) Partner in Crime

Introduction

- How you doin’

About Me

- Nadeem Douba
  - Work at Cygnos (http://www.cygnos.com) in Ottawa, ON, Canada
  - Certs: GWAPT, GPEN
  - Worked in the InfoSec field for 10+ years.
  - Love (European) football and hacking stuff…
- Been a Maltego fan-boy since the beginning…
- Helped port/appify Maltego for Mac OS X ☺

Presentation Overview

- What is Sploitego?
- Maltego – Briefly Explained
- Dive Into Development
  - Before Sploitego
  - After Sploitego
- Demos
- Conclusion
- Questions

What is Sploitego?

- Local Transform Development Framework for Maltego written in **Python**
- Provides:
  - Rapid transform development
  - Easy transform installation, management, and maintenance
  - Complementary scripts and modules for data mining and debugging
  - A whole bunch of cool transforms ☺

But First…

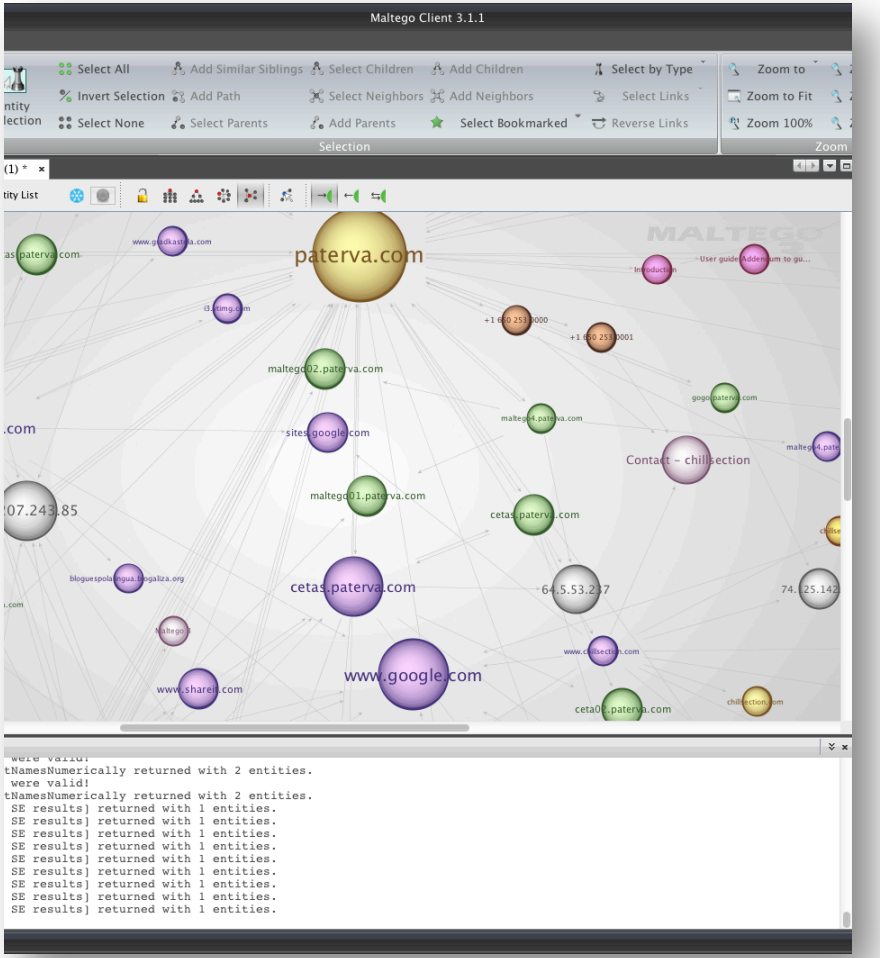
- A little background on Maltego…

Background

- Maltego Overview

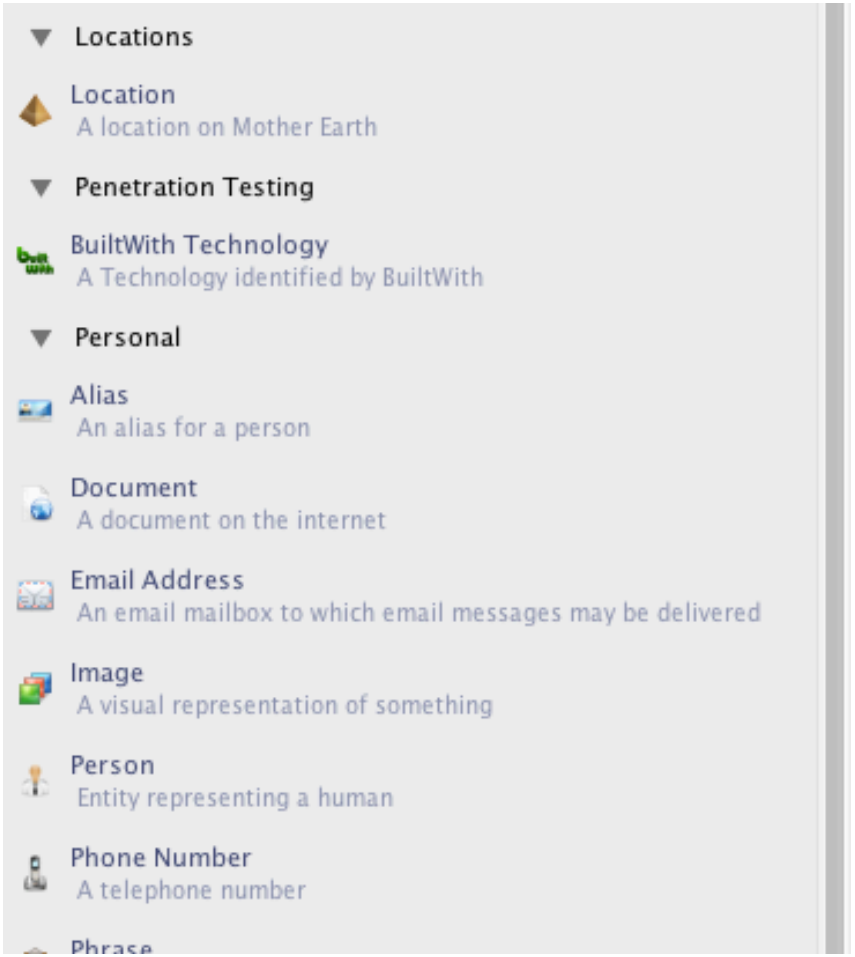
What is Maltego?

- Open Source Intelligence (OSInt) and forensics information mining/gathering and graphing tool
- Developed by *Paterva* and *PinkMatter*

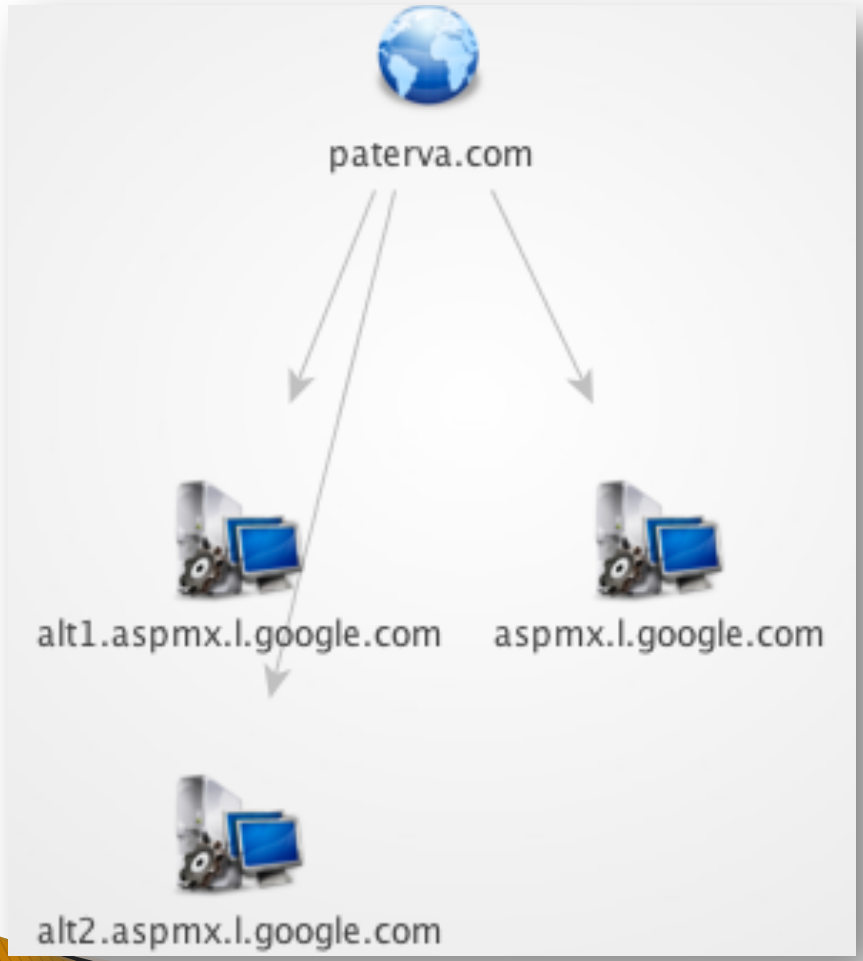


What is Maltego? – Cont.

- Information is represented on the graph as **Entities**
- For example, an **Entity** could be:
  - Email Address
  - Image
  - Phone Number
  - Etc.
- Each **Entity** has a value and optionally some fields.



What is a Transform?



- **Transforms** reveal relationships between **entities** (or information)
- Logic that mines and returns information (or **Entities**) using another piece of information (or **Entity**) as input
  - $T(E_0) \rightarrow \{ E_1, E_2, \ldots E_n \}$

Maltego Demo

- Just for Clarity

What is a Transform? – Cont.

- **Two** types of transforms:
  - **Remote**: runs on a remote Paterva or third-party *Transform Server*.
  - **Local**: runs on the user's local machine.
    - This is where **Sploitego** comes in…

Remote Transforms – Pros & Cons

- ✓ Paterva's Transforms
  - ✓ Awesome!!!
- ✓ Centralized Transform Management & Maintenance
  - ✓ Implementation details hidden from the user (protects your IP)
- ✓ Minimal Client-Side Processing Overhead

Pros

- ✗ Limited Data Visibility
  - ✗ i.e. Server can only query accessible data.
- ✗ Breach of Privacy
  - ✗ OSInt target/subject disclosed to a third-party.
- ✗ Limited Client-Side Control:
  - ✗ Transforms might not be evil enough ☺

Cons

Local Transforms – Pros and Cons

- ✓ Full Client-side Control
  - ✓ No limits as to how 1337 or evil your transforms can be ☺
- ✓ Privacy
  - ✓ OSInt subject may not be disclosed to third-party
- ✓ Great Data Visibility
  - ✓ "The world is one's oyster"
- ✓ Extensible
  - ✓ Maltego can be used for other types of data visualization ☺

Pros

- ✗ Processing Overhead
  - ✗ Client's machine responsible for running transforms
- ✗ Development
  - ✗ It's in your hands (or somebody else's… just delegate ;)
- ✗ IP Disclosure
  - ✗ Implementation details no longer hidden from users.
- ✗ Difficult to Maintain

Cons

Local Transform Development

- The Nitty Gritty

How do Local Transforms Work?

- Maltego executes a local script or executable
- Input passed via command line arguments:

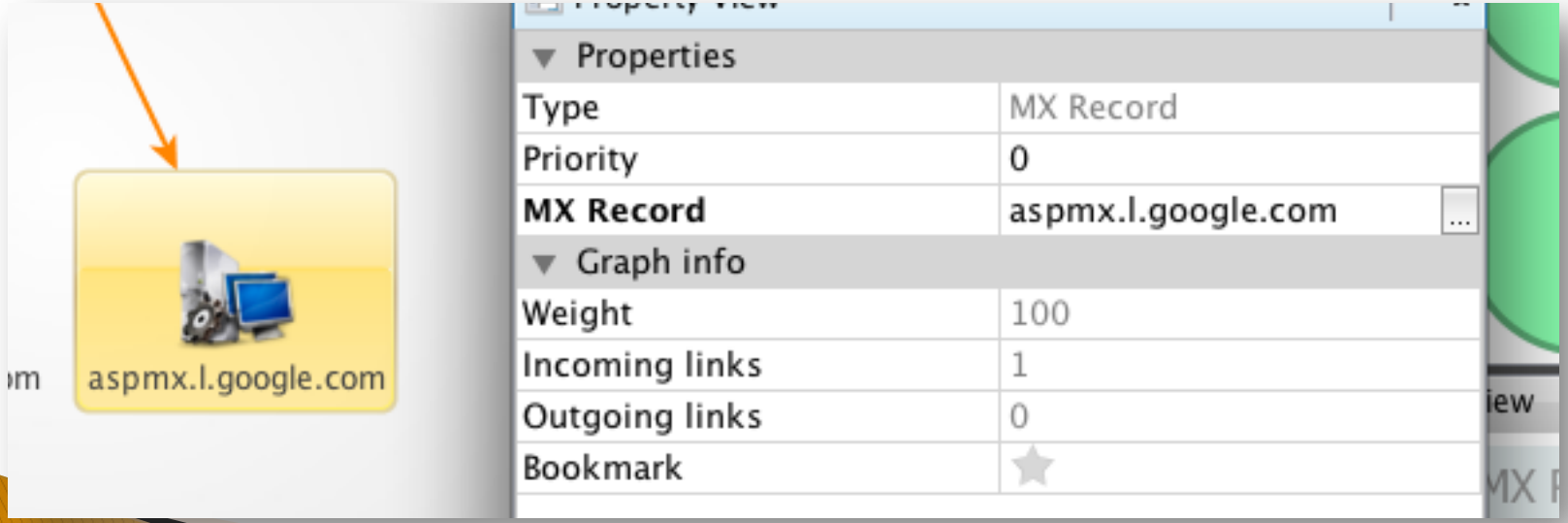
  ```
  $ ./mytransform.sh <value>
    <fieldname1>=<fieldvalue1>#...#<fnn>=<fvn>
  ```

- Transform results returned via standard output in Maltego XML message format
  - **See:** http://paterva.com/web5/documentation/localTransforms-SpecIII.pdf for more details
- Debugging messages returned via standard error

Example – Transform Call

```
$ ./t.pl aspmx.l.google.com
mxrecord.priority=0
```

- **Note:** the bolded property (“MX Record”), below, is the entity value (or Display Value)



Example – Transform Message

```
altegoMessage>
  <MaltegoTransformResponseMessage>
    <Entities>
      <Entity Type="maltego.IPv4Address">
        <Value>0.0.0.0</Value>
        <Weight>1</Weight>
        <AdditionalFields>
          <Field DisplayName="Internal"
                 MatchingRule="strict"
                 Name="ipaddress.internal">true</Field>
          <Field DisplayName="Hardware Address"
                 MatchingRule="strict"
                 Name="ethernet.hwaddr">00:00:00:00:00:00</Fiel
        </AdditionalFields>
      </Entity>
    </Entities>
  </MaltegoTransformResponseMessage>
MaltegoMessage>
```

Writing a Local Transform

- Without Sploitego

Local Transform Development Checklist

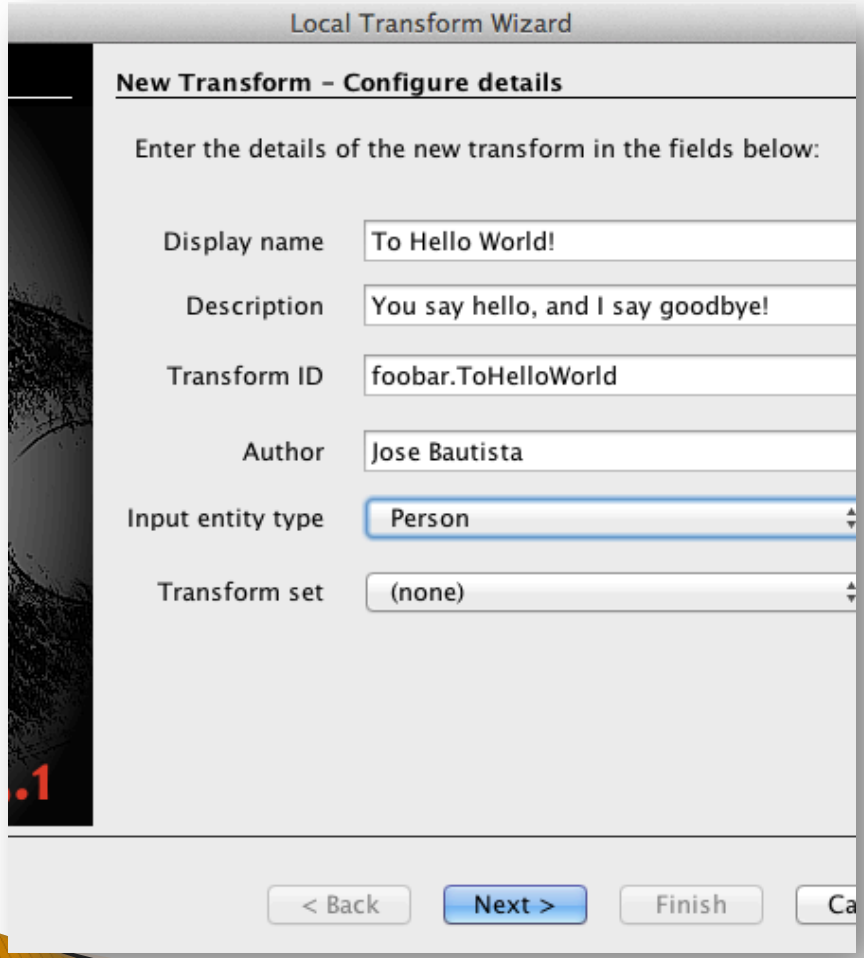
- Learn Maltego Local Transform Specification
  - XML Messaging
  - Debugging
  - Etc.
- Develop Transform
  - Input Parsing Logic
  - Data Mining Logic
  - XML Serialization Logic
  - Debugging Facilities
- Install Transform
- Configure & Maintain Transform
- Define Entity in Maltego (Optional)

Example – Hello World Transform

```
#!/usr/bin/env python

from sys import exit, argv, stderr
from re import split

def parseargs(args=argv):
    """Parse arguments for Maltego local transforms."""

    if len(args) < 3:
        stderr.write('usage: %s <transform> [param1 ... paramN] <value> [field1=value
        exit(-1)

    arg_script = args[1]
    arg_field = args[-1] if '=' in args[-1] else None
    arg_value = args[-1] if arg_field is None else args[-2]
    arg_param = []

    if arg_field is None and len(args) > 3:
        arg_param = list(args[2:-1])
    elif arg_field is not None and len(args) > 4:
        arg_param = list(args[2:-2])

    fields = {}
    if arg_field is not None:
        fs = split(r'(?<=[^\\])#', arg_field)
        if fs is not None:
            fields = dict(map(lambda x: x.split('=', 1), fs))

    return arg_script, arg_param, arg_value, fields


def debug(*args):
    """Send debug messages to the Maltego console."""
    for i in args:
        stderr.write('D:%s\n' % str(i))


if __name__ == '__main__':
    args, params, value, fields = parseargs()
    debug('Running Hello World! Transform')
    print '<MaltegoMessage><MaltegoTransformResponseMessage>' \
          '<Entities><Entity Type="maltego.Person">' \
          '<Value>Hello %s</Value><Weight>1</Weight>' \
          '<AdditionalFields></AdditionalFields>' \
          '</Entity></Entities></MaltegoTransformResponseMessage>' \
          '</MaltegoMessage>' % value
    exit()
```

- 47 lines of code for a simple transform
  - Not bad…
  - But not great either
- XML is hard-coded
  - Not reusable
  - Debugging nightmare!
  - Imagine returning 100+ entities with fields 😐

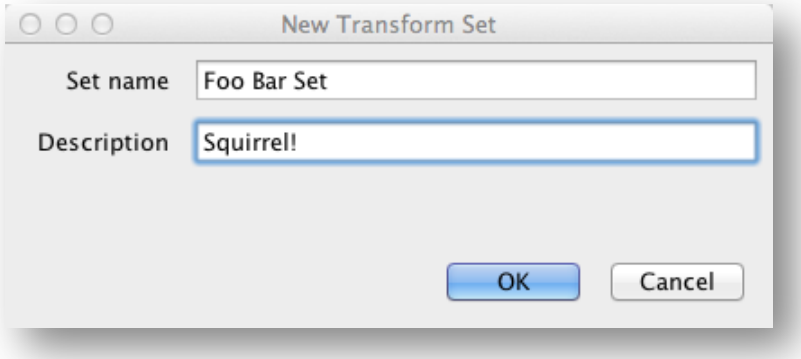
Installing Transforms



- Currently Manual Process
  - Two-step Wizard per Transform
- Tedious & Prone to User Error
  - More Transforms = More Configuration = Less Time Playing

Grouping Transforms

- Have to manually create a *Transform Set*
- Another dialog box somewhere 😐
- When does the fun begin?



Sploitego

- Bringing Back the Fun

What is Sploitego?

- Local Transform Development Framework for Maltego written in *Python*
- Provides:
  - Rapid transform development
  - Easy transform installation, management, and maintenance
  - Complementary scripts and modules for data mining and debugging
  - A whole bunch of cool transforms ☺
- How does it bring back the fun?

Remember our Checklist?

- ✓ Learn Maltego Local Transform Specification
  - ✓ XML Messaging
  - ✓ Debugging
  - ✓ Etc.
- ✓ Develop Transform
  - ✓ Input Parsing Logic
  - ◦ Data Mining Logic ← **This is all you have to take care of! – Wawa-wiwa!**
  - ✓ XML Serialization Logic
  - ✓ Debugging Facilities
- ✓ Install Transform
- ✓ Configure & Maintain Transform
- ▸ Define Entity in Maltego (Optional) ← **And possibly this…**

Sploitego Transforms – Packaging

- Sploitego transforms are simply **Python Modules** within **Python Packages**
- Follows traditional Python package directory structure:
  - `./setup.py` (Python installation script – distutils/ setuptools)
  - `./foobar` (Package directory)
  - `./foobar/__init__.py` (Module/package init script)
  - `./foobar/helloworld.py` (Transform module)

Hello World (Revised) Transform

```
3   from sploitego.maltego.message import Person, Phrase
4   from sploitego.maltego.utils import debug, progress
5   from sploitego.framework import superuser, configure
6
7   @superuser
8   @configure(
9       label='To Phrase [Hello World]',
10      description='Returns a phrase entity with the phrase "Hello Word!"',
11      uuids=[ 'sploitego.v2.PersonToPhrase_HelloWorld' ],
12      inputs=[ ( 'Useless', Person ) ],
13      debug=True
14  )
15  def dotransform(request, response):
16      progress(50)
17      debug('This was pointless!')
18      progress(100)
19      return response + Phrase('Hello %s' % request.value)
20
21
22  def onterminate():
```

Additional Steps

- `foobar/__init__.py` must contain `__all__` variable with transform modules specified.

```
1  #!/usr/bin/env python
2
3  __all__ = [
4      'helloworld'
5  ]
```

Dissecting the Transform

Sploitego Transform – Dissected

- The `dotransform` function is the entry point
- Accepts two parameters: `request`, and `response`
- The `request` object has the following properties:
  - `value`: the Entity display value (string)
  - `fields`: the Entity fields (dictionary)
  - `params`: extra parameters that can be parsed by `optparse`

Sploitego Transform – Dissected – cont.

- The `response` object is where we populate our results
- `dotransform` must return the `response` object
- `response` object uses mathematical operators to add and remove `Entity` and `UIMessage` objects
  - E.g. `response + Phrase('Hi')` appends a Phrase Entity object to the response object
- Finally, `onterminate` function is called if *Maltego* interrupts the transform – it is optional

Transform Execution – Meta-data

- @superuser instructs the dispatcher to run the transform as the super-user
- If a transform is being executed as a non-super-user:
  - dispatcher will invoke sudo
  - Prompt user for sudo password
  - If successful, execute the transform using sudo
  - Else, abort execution after three retries

Installation Meta-Data – *@configure*

```
8   @configure(
9       label='To Phrase [Hello World]',
10      description='Returns a phrase entity with the phrase "Hello Word!"',
11      uuids=[ 'sploitego.v2.PersonToPhrase_HelloWorld' ],
12      inputs=[ ( 'Useless', Person ) ],
13      debug=True
14  )
```

- Instructs `mtginstall` on how to configure transform in Maltego
- **Parameters**:
  - **`label`**: display label of transform in Maltego
  - **`description`**: A brief description
  - **`uuids`**: list of universally unique identifiers (or transform descriptor file names)
  - **`inputs`**: list of tuples containing *Transform Set* name and *Input Entity* type
  - **`debug`**: whether or not debug window should appear in Maltego on transform execution

Installation Meta-Data – *@configure* – cont.

```
8   @configure(
9       label='To Phrase [Hello World]',
10      description='Returns a phrase entity with the phrase "Hello Word!"',
11      uuids=[ 'sploitego.v2.PersonToPhrase_HelloWorld' ],
12      inputs=[ ( 'Useless', Person ) ],
13      debug=True
14  )
```

- For example:
  - Transform will appear as `To Phrase [Hello World]` in Maltego GUI
  - Will belong to the `Useless` Transform Set
  - Can only be applied to *Person* type Entities
  - Have a unique ID of `sploitego.v2.PersonToPhrase_HelloWorld`
  - A debug window will appear on transform execution

Installation Meta-Data – *@configure* – One more Thing.

- Notice how **uuids** and **inputs** are lists
- `mtginstall` supports one-to-many relationship between transforms and input entity types
  - For example, *Hello World Transform* could be applied to *Phrase* entities as well
  - Just add another **uuid** and **inputs** entry (matching order)

```
@configure(
    label='To Phrase [Hello World]',
    description='Returns a phrase entity with the phrase "Hello Word!"',
    uuids=[ 'sploitego.v2.PersonToPhrase_HelloWorld',
            'sploitego.v2.PhraseToPhrase_HelloWorld' ],
    inputs=[ ( 'Useless', Person ),
             ( 'Useless', Phrase ) ],
    debug=True
)
```

Hello World (Revised) – The Stats

- **24 Lines of Code in Total!**
  - Approximately 50% less code!
  - Only **SIX (6)** lines were "actual" code!
  - The rest were annotations, function signatures, and imports
- Not a single print line in sight!
- No hard-coded XML!
- What about installation?

Managing Transform Packages

- Install, Uninstall, Etc.

Installing Transforms (Revised)

- To install a Sploitego transform:
  - **First,** Install Python package containing transforms
    - distutils or setuptools are great for that!
  - **Alternatively,** place Python module in Maltego's working directory
  - **Second,** run `mtginstall`

Installing Transforms (Revised) – Cont.

- **Input Parameters:**
  - *Hello World Transform* is in `foobar` package
  - Maltego's settings are stored in `~/Library/Application\ Support/maltego/v3.1.1/` (on Mac OS X)
  - Your transform working directory is `~/`
- **To Install Transform Package, Run:**

```
$ mtginstall --package foobar --maltego-prefix
  ~/Library/Application\ Support/maltego/v3.1.1/
  --working-dir ~/
```

Transform Installer – *mtginstall*

1. `mtginstall` first imports `__init__.py` in `foobar` package
2. Iterates the `__all__` special variable to get list of modules in package
3. Loads each module and looks for `dotransform` function annotated with `@configure`
4. Reads installation meta-data and installs transform in Maltego accordingly
   a) If *Transform Set* doesn't exist, it will create it.
   b) Detects name collisions between transforms

Uninstalling Transforms

- To uninstall a Sploitego transform run `mtguninstall`:

```
$ mtguninstall --package foobar --maltego-prefix
  ~/Library/Application\ Support/maltego/v3.1.1/
```

- `mtguninstall` will remove the transform package (Transform Sets and Transforms) from Maltego's GUI but not from Python site-package directory

Demos

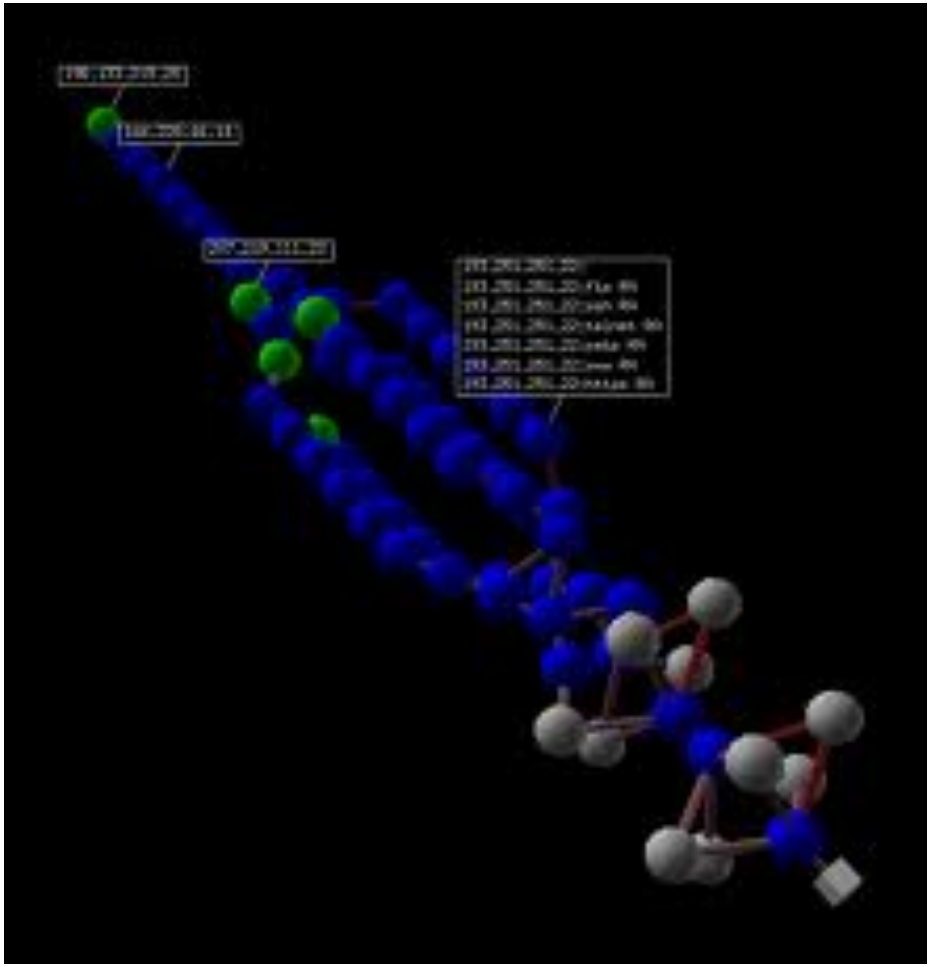
- The Fun Stuff

Metasploit Integration

Demo

Nmap/Amap Integration

Demo



Scapy Integration

Demo

Extra Utilities

- The Goodies

Debugging, Testing, Etc.

- `mtgdebug` script prints results in readable format
- `mtgsh` shell version of `mtgdebug` – still a work in progress

```
bitter:~ ndouba$ mtgdebug sploitego.transforms.whatismyip -
`- MaltegoTransformResponseMessage:
  `- Entities:
    `- Entity:  {'Type': 'maltego.IPv4Address'}
      `- Value: 0.0.0.0
      `- Weight: 1
      `- AdditionalFields:
        `- Field: true {'DisplayName': 'Internal', 'Name': 'ipaddress.i
nternal', 'MatchingRule': 'strict'}
        `- Field: 00:00:00:00:00:00 {'DisplayName': 'Hardware Address',
 'Name': 'ethernet.hwaddr', 'MatchingRule': 'strict'}
```

Graph Export Conversion Tools

- `mtgx2csv` converts exported Maltego graphs to CSV (comma-separated value) format.
- `csv2sheets` reads the output of `mtgx2csv` and separates entities of the same type into separate CSVs

CONCLUSIONS

- Last but not Least

Project Roadmap

- Get a website up with some **documentation** ☺
- Create more transforms for:
  - Social Engineering
  - Forensics
  - Exploitation
  - Scanning and Vulnerability Discovery
  - Third-party Tool Integration
  - Etc.
- Create an online community and transform package index for transform developers similar to PyPI
- Develop a context engine
  - Minimize data duplication on graphs
  - Provide transforms with access to full graph

Looking for Help!

- Sploitego needs your help!
  - Developers
  - Transform Gurus
  - Hackers
  - Documenters
  - Website Designers
  - Chefs who deliver to the Ottawa area☺

Contact Info

- Please feel free to contact me:
  - Email: ndouba@gmail.com
  - Twitter: @ndouba
  - Skype: nadeem.douba

Kudos

- To the Paterva team:
  - Andrew MacPherson (Mohawk)
  - Roelof Temmingh (RT)
- To the Cygnos & RCGT team (w00t!)
- Thank you for attending!

Questions

- Anyone?